Contents
ESSENTIAL INFORMATION .................................................................................................................................................... 3
VARIABLE DEFINITIONS AND VARIABLE CALLING ....................................................................................................... 3
ALGORITHM DESIGNER .......................................................................................................................................................... 4
BASIC COMMANDS .................................................................................................................................................................  4
GOOD PROGRAMMING PRACTICE .................................................................................................................................... 6
   USING COMMENTS IN YOUR CODE.............................................................................................................................. 6
   FORWARD THINKING.................................................................................................................................................... 6
COMMON TASKS ..................................................................................................................................................................... 7
   GENERATE RANDOM NUMBERS TO USE IN A FORMULA ......................................................................................... 7
   FORCE NUMBERS TO DISPLAY IN A PARTICULAR WAY ................................................................................. 7
   GENERATE RANDOM TEXT ITEMS FROM A LIST OF ITEMS ............................................................................. 7
   CONVERT FROM DEGREES TO RADIANS TO PERFORM A TRIG CALCULATION .................................... 7
   PERFORM A TRIG CALCULATION THEN CONVERT THE ANGLE TO DEGREES ........................................ 7
USING IF STATEMENTS ........................................................................................................................................................... 8
   TEXT BASED IF STATEMENT ........................................................................................................................................ 8
   CHECK IF A NUMBER IS ODD OR EVEN ................................................................................................................... 8
USING A SWITCH-FUNCTION .............................................................................................................................................. 8
USING CONDITIONS ................................................................................................................................................................. 9
THE MAPLE FUNCTION........................................................................................................................................................ 11
   USING A SINGLE CALL FOR MANY CALCULATIONS .................................................................................. 11
   ADDING PHASORS/VECTORS (INCLUDING CONVERTING DEGREES TO/FROM RADIANS)........ 12
GRAPHING AND PLOTTing .................................................................................................................................................... 13
FORCING THE DISPLAY OF NATURAL MATHEMATICS ................................................................................... 15
WORKING WITH MATHEMATICAL FUNCTIONS ........................................................................................................ 15
LOOKUP VALUES IN A TABLE.............................................................................................................................................. 16
ESSENTIAL INFORMATION

MapleTA utilizes two primary methods to assist in the authoring of algorithms:

i. The utilization of simple to use built-in functions with common crossover to various functions commonly used in HTML5 programming.

ii. Utilization of a computer-algebra-system to perform calculations that are significantly more labour intensive.

There are a wide range of uses to building algorithms using MapleTA, with enough practice, beginner to advanced users, will find the platform quite user friendly and a definite time-saver when building randomized questions.

Most users will never use the computer-algebra-system component of MapleTA, as such all users are strongly encouraged to become familiar with the standard built-in functions available in the system prior to authoring questions using the algebra-system. For updated and extensive help regarding the standard built-in functions visit:


This page can be found via a Google search for “MapleTA builtin”.

This support document will outline various programming techniques and syntax commonly used when working with algorithms in MapleTA.

VARIABLE DEFINITIONS AND VARIABLE CALLING

Variables are defined and referenced (throughout question text, answer fields and algorithms) using a “$” precursor, and must not contain spaces or special characters. $variableWilliam, $firstNum, and $currentTopBranch are all examples of properly defined and referenced variables.

NOTICE: We recommend you use camelCase naming by separating words using a capital letter at the start of each. This altering between upper and lower case makes reading the phrase much easier, given spaces are not allowed. For example:

$thisVariableWillBeTheFirstNumberInMyAlgorithm

IMPORTANT: Because the $ is used as a precursor to a randomized variable, the currency symbol, $, can be difficult to display on the screen. To display the currency symbol you must use the “\” escape character, prior to typing the variable name. For example to display a randomized variable entitled “$accountTotal” we would type “\$accountTotal” in the question text.
ALGORITHM DESIGNER

MapleTA includes a simplified “Algorithm Designer” that allows users to create randomized questions without worrying about proper syntax. To get started with the designer click “Show Designer” within the algorithm tab.

For example:
Suppose you want a random number between 5 and 10 generated in steps of 1, as in:

\[ \text{Num1} = \text{one of the following numbers: 5, 6 7, 8, 9, 10} \]

We can use the designer, as shown below.

Suppose we want \( \text{num2}=2\times\text{num1} \) (when \( \text{num1} \) is less than 8) and \( \text{num2}=3\times\text{num1} \) otherwise.

As noted above the algorithm designer is a great way to become familiar with various syntax in the system. Over time as authors become more proficient it is likely they will adopt to programming their code directly into the algorithm text-area.

BASIC COMMANDS

As previously mentioned MapleTA has a large number of smaller, simple-to-use, commands for randomization as well as basic calculation. A list of commands and basic instructions pertaining thereto can be located at:

This page can be found via a Google search for “MapleTA builtin”.

For a list of sample commands see the next page.
<table>
<thead>
<tr>
<th>Command</th>
<th>What does this command do?</th>
<th>Sample (<em>special notes</em>)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$var1=range (&lt;a&gt;,&lt;b&gt;,&lt;c&gt;);</td>
<td>Generates a random number between &lt;a&gt; and &lt;b&gt; in steps of &lt;c&gt;.</td>
<td>$var1=range(-5,7,2); *notice this cannot ever produce 0.</td>
</tr>
<tr>
<td>$number2=rint(&lt;b&gt;);</td>
<td>Generates a random number from 0 to &lt;b&gt;, not including &lt;b&gt; itself.</td>
<td>$number2=rint(17); *notice this cannot ever produce 17.</td>
</tr>
<tr>
<td>$markNum=rand(&lt;a&gt;,&lt;b&gt;); or $markNum=rand(&lt;a&gt;,&lt;b&gt;,&lt;c&gt;);</td>
<td>Generates a random number between &lt;a&gt; and &lt;b&gt;, with optional &lt;c&gt; significant figures.</td>
<td>$markNum=rand(2.4155,57.6); *notice this will force the display of 6 significant figures.</td>
</tr>
<tr>
<td>$billRound=decimal(&lt;a&gt;,&lt;?&gt;);</td>
<td>The number &lt;?&gt; is rounded to &lt;a&gt; decimal places.</td>
<td>$billRound=decimal(3,45,98776); *notice this equals 45.988</td>
</tr>
<tr>
<td>$forceDisp=numfmt(&quot;___&quot;,&lt;a&gt;);</td>
<td>Force a specific display of &lt;a&gt;</td>
<td>$forceDisp=numfmt(&quot; 00.000&quot;,1.2); *notice this results in 0.1200</td>
</tr>
<tr>
<td>$daveForce=sign(&lt;a&gt;,&lt;b&gt;);</td>
<td>Force &lt;b&gt; to display to &lt;a&gt; sig-digs.</td>
<td>$daveForce=sign(5,12); *notice this equals 3.1416</td>
</tr>
<tr>
<td>$physics=lsu(&lt;a&gt;,&lt;b&gt;);</td>
<td>Computes the unit in the &lt;a&gt; significant place of the number &lt;b&gt;.</td>
<td>$physics=lsu(4,779688); *notice this equals 100</td>
</tr>
<tr>
<td>$scottNum=int(&lt;a&gt;);</td>
<td>Force the number &lt;a&gt; to round DOWN to nearest whole number (floor function).</td>
<td>$scottNum=int(7.999); *notice this equals 7</td>
</tr>
<tr>
<td>$alexNum=max(&lt;a&gt;,&lt;b&gt;,&lt;c&gt;); or $alexNum=min(&lt;a&gt;,&lt;b&gt;,&lt;c&gt;);</td>
<td>Returns the largest (or smallest) of the numbers.</td>
<td>$alexNum=max(3,7,-12); *notice this equals 7</td>
</tr>
<tr>
<td>$forcePos=abs(&lt;a&gt;);</td>
<td>Forces the number to be a positive (absolute value).</td>
<td>$forcePos=abs(-12); *notice this equals 12</td>
</tr>
<tr>
<td>$fancyDis=frac(&lt;a&gt;,&lt;b&gt;);</td>
<td>Forces the display of &lt;a&gt;/&lt;b&gt; as opposed to the decimal equivalent.</td>
<td>$fancyDis=frac(1.3); *displays “1/3” as opposed to “0.333”</td>
</tr>
<tr>
<td>$helloNum=act(&lt;a&gt;);</td>
<td>Computes &lt;a&gt; factorial.</td>
<td>$helloNum=act(4); *notice this equals 24</td>
</tr>
<tr>
<td>$combinatorics=binomial(&lt;a&gt;,&lt;b&gt;);</td>
<td>Computes ( C_{&lt;b&gt;}^{&lt;a&gt;} )</td>
<td>$combinatorics=binomial(5,2); *notice this equals 10</td>
</tr>
<tr>
<td>$commonDiv=gcd(&lt;a&gt;,&lt;b&gt;);</td>
<td>Computes the greatest common divisor between &lt;a&gt; and &lt;b&gt;</td>
<td>$commonDiv=gcd(18,24); *notice this equals 6</td>
</tr>
<tr>
<td>$position=indexof(&lt;&lt;?&gt;,&lt;a&gt;,&lt;b&gt;,&lt;c&gt;,..);</td>
<td>Computes the position of &lt;?&gt; in the list &lt;a&gt;,&lt;b&gt;,&lt;c&gt;,.. (note &lt;a&gt; is considered the 0 position).</td>
<td>$position=indexof(5,6,2,1,9,5,9); *notice this equals 2</td>
</tr>
<tr>
<td>$positionR=rank(&lt;&lt;?&gt;,&lt;a&gt;,&lt;b&gt;,&lt;c&gt;,..);</td>
<td>Computes the position of the &lt;?&gt; largest item in the list.</td>
<td>$positionR=rank(3,8,7,-4,91); *notice this equals -4</td>
</tr>
<tr>
<td>$commonDiv=gcd(&lt;a&gt;,&lt;b&gt;);</td>
<td>Computes the greatest common divisor between &lt;a&gt; and &lt;b&gt;</td>
<td>$commonDiv=gcd(18,24); *notice this equals 6</td>
</tr>
<tr>
<td>$stats1=erf(&lt;a&gt;);</td>
<td>Computes the probability that a value attains a z-score less than or equal to &lt;a&gt;.</td>
<td>$stats1=erf(2); *notice this equals 0.97725</td>
</tr>
<tr>
<td>$stats2=inverf(&lt;a&gt;);</td>
<td>Computes the value of a z-score that correlates with the given probability &lt;a&gt;.</td>
<td>$stats1=inverf(0.97725); *notice this equals 2.00</td>
</tr>
<tr>
<td>$cumul=sum(i,&lt;a&gt;,&lt;b&gt;,2^i);</td>
<td>Computes ( \sum_{i=a}^{b} 2^i )</td>
<td>$cumul=sum(1,3,12); *notice this equals 14</td>
</tr>
<tr>
<td>$displayFancy=mathml(&lt;a&gt;);</td>
<td>Produces fancy mathematics display for the expression &lt;a&gt;.</td>
<td>$displayFancy=mathml(1+3<em>sqrt(2</em>x));</td>
</tr>
<tr>
<td>$stringWork=strcat(&quot;&lt;a&gt;&quot;,&quot;&lt;b&gt;&quot;);</td>
<td>Concatenates &lt;a&gt; with &lt;b&gt;</td>
<td>$stringWork=strcat(&quot;red&quot;,&quot;Green&quot;); *notice this equals “redGreen”</td>
</tr>
</tbody>
</table>

NOTICE: Commands can be embedded within one another. Such as:
\$numberMark=decimal(2,range(4,18,0.0001));
The above code runs from the inside out, first generating a random number (using range), then rounding the number to 2 decimal places.
GOOD PROGRAMMING PRACTICE

USING COMMENTS IN YOUR CODE
If you place a hashtag at the start of a line in your algorithm the line will be “commented out”, in other words hashtags are used as notes to the author rather than actual syntax for the computer to read.

For example:

```
$randomNum1=range(-10,10,1);  //author comment: this line generates a number between -10 and 10.
$randomNum2=range(2,7,1);     //this line generates a number between 2 and 7.
```

IMPORTANT: Comments should be used to:
- help other authors read and understand your algorithm.
- help clarify something you may have written a long time ago.
- troubleshoot by having the system ignore problematic lines of code rather than deleting them.

Suppose you have a complex question of which the answer, with solution, is provided on scrap paper, the question author can use comments to block out range commands and subsequently test the algorithm using the single static case.

For example:

```
#we will be using hashtags to ignore range commands until we are sure the calculation is correct/working.
$resistance1=6;  //range(-10,10,1);
$resistance2=8;  //range(-10,10,1);
$resistanceTotal=1/(1/$resistance1) + (1/$resistance2));
```

FORWARD THINKING
Quite often too much randomness in algorithms can cause problems.

For example:

```
$var1=range(-10,10,1);
$var2=range(-10,10,1);
$var3=range(-10,10,1);
$var4=range(-10,10,1);
$answer=(1/$var1)+ (1/$var2); //add the fractions
```

The above algorithm will occasionally produce: \( \frac{3}{0} + \frac{-8}{-2} \) which could be viewed as having several problems. For the division by zero problem consider one of the alternatives below.

**Option 1**
```
$var2=range(9,9,2);
$var4=range(9,9,2);
#the above code given the steps of 2 will skip over zero.
```

**Option 2**
```
$var2=switch(rint(2), -1, 1)*range(1,9,1);
$var4=switch(rint(2), -1, 1)*range(1,9,1);
#the above code will take either +1 or -1 and multiply by the range.
```
COMMON TASKS

GENERATE RANDOM NUMBERS TO USE IN A FORMULA

\$time=\text{range}(200,500,1); \\
\$velocity=\text{decimal}(3,\text{range}(4.5,7.8,0.0001)); \#\text{force a display rounded to 3 decimal places} \\
\$answer=\$velocity*\$time; \#\text{calculate the distance travelled}

FORCE NUMBERS TO DISPLAY IN A PARTICULAR WAY

\$currency=\text{range}(1,5,0.01); \#\text{this could produce the currency value 5.1 which we might want to display as 5.10} \\
\$numberFormat=\text{numfmt}(\"\#.00\",\$currency); \#\text{this will force the display to two decimal places} \\
\$phoneNumber=\text{range}(1000000,9999999,1); \#\text{this will produce a 7-digit number} \\
\$numberFormat2=\text{numfmt}(\"7800000000\",\$phoneNumber); \#\text{this will force a 780 at the start of the number} \\
\$physics=\text{range}(5,12,0.5); \\
\$numberFormat3=\text{sig}(3,\$physics); \#\text{this will force 3 significant digits}

NOTICE: Several commands can be embedded within one another. Such as:

\$numberMark=\text{decimal}(2,\text{range}(4,18,0.0001));

GENERATE RANDOM TEXT ITEMS FROM A LIST OF ITEMS

\$list=\text{\"fast rabbit\",\"delta\",\"slow fox\",385.72,\"sqrt(6)\"}; \\
\#\text{notice- the list is encased in single-quotes, while separate phrases (items) are encased in double-quotes.} \\
\$itemPulledFromList=\text{switch}(\text{rint}(5),\$list); \\
\#\text{notice- we are using rint(5) to pull a random element from \$list which has \"5\" elements.} \\
\#\text{below is an alternate method} \\
\$item=\text{switch}(\text{rint}(5),\"fast rabbit\",\"delta\",\"slow fox\",385.72,\"sqrt(6)\"); 

CONVERT FROM DEGREES TO RADIANS TO PERFORM A TRIG CALCULATION

\$angleDeg=\text{range}(0,360,30); \#\text{generate a random angle from 0 to 360 in steps of 30} \\
\$angleRad=\$angleDeg*(\text{Pi}/180); \#\text{convert the angle to radians} \\
\$value=\text{sin}(\$angleRad); \#\text{calculate the sin of the angle}

PERFORM A TRIG CALCULATION THEN CONVERT THE ANGLE TO DEGREES

\$angleRad=\text{asin}(0.5); \#\text{use the inverse sine function to find an angle} \\
\$angleDeg=\$angleRad*(180/\text{Pi}); \#\text{convert the angle to degrees}

NOTICE: The algebra-system (explained later) can be utilized to simplify conversion from degrees to radians.
USING IF STATEMENTS

IMPORTANT: For very complex conditional statements a switch function or a matrix should be used as opposed to an if-statement.

If-statements rely on a boolean-logic test, followed by a “value if true” statement, and then followed by a “value if false” statement, for example:

\[
\text{$myVariable= if( <a> , <b> , <c>); #whereby <a>=boolean test, <b>=value if <a> is true, and <c> value if <a> is false}
\]

In other words you are going to have to familiarize yourself with Boolean-Logic in order to write if-statements.

<table>
<thead>
<tr>
<th>Boolean statement</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>eq(3,3.0)</td>
<td>True</td>
</tr>
<tr>
<td>ne(5,5);</td>
<td>False</td>
</tr>
<tr>
<td>gt(3,4);</td>
<td>False</td>
</tr>
<tr>
<td>lt(3,4);</td>
<td>True</td>
</tr>
<tr>
<td>ge(0,0);</td>
<td>True</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Boolean statement</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>eq(&quot;green&quot;,&quot;Green&quot;)</td>
<td>False</td>
</tr>
<tr>
<td>ne(3.14,Pi);</td>
<td>False</td>
</tr>
<tr>
<td>gt(3.00,3);</td>
<td>True</td>
</tr>
<tr>
<td>lt(3,abs(-4));</td>
<td>True</td>
</tr>
<tr>
<td>le(-2,-3);</td>
<td>False</td>
</tr>
</tbody>
</table>

TEXT BASED IF STATEMENT

\[
\text{$volt= switch(rint(2), 110 , 220); #generate the number 110 or 220}$
\]

\[
\text{$isItHighVoltage= if( lt($volt, 150), "low" , "high"); #if $volt is less than, "lt", 150-> "low", otherwise-> "high"}$
\]

CHECK IF A NUMBER IS ODD OR EVEN

\[
\text{$num=range(200 , 500 , 1); #generate a random number}$
\]

\[
\text{$oddOrEven=if( eq( \gcd($num , 2) ,2), "even","odd"); #if the gcd of ($num and 2) is 2 -> even, otherwise -> odd}$
\]

USING A SWITCH-FUNCTION

Suppose you have a question that asks students to determine the time it takes to travel to a destination based on the table below. Perhaps the data values are placed directly into the context of the question, rather than supplying students with a table. This approach dynamically randomizes questions while ensuring the various versions correspond with identical learning outcomes.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Calgary</th>
<th>Toronto</th>
<th>Kingsway Mall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method of Travel</td>
<td>drive</td>
<td>fly</td>
<td>walk</td>
</tr>
<tr>
<td>Distance (km)</td>
<td>322</td>
<td>2713</td>
<td>1.2</td>
</tr>
<tr>
<td>Speed (km/hr)</td>
<td>100</td>
<td>800</td>
<td>5.0</td>
</tr>
</tbody>
</table>

We outline the applicable question text and algorithm on the next page.
Question Context:

Jimmy is to $methodOfTravel from NAIT to $destination, a distance of $distance km. Determine the time the journey would take provided he is to travel at a speed of $speed km/h?

Answer= $answerTime

Algorithm:

$version= rint(3);
#3 versions because there are 3-methods of travel
$destination= switch($version, "Calgary" , "Toronto" , "Kingsway Mall");
#notice. version0=Calgary, version1=Toronto, version2=Kingsway Mall
#notice... the list must always occur in the corresponding order to the columns in the table
methodOfTravel= switch($version, "drive" , "fly" , "walk");
$distance= switch($version, 322 , 2713 , 1.2);
$speed= switch($version, 100 , 800 , 5.0);
$answerTime= $distance/$speed;

In the algorithm below students will be presented with a random Trigonometric Function (sine, cosine, or tangent) and asked to determine the function value.

Question Context:

Determine the value of $trigFunction($angleDeg°)?

Round your answer to 2-decimal places.

Answer= $answerRound

Algorithm:

$angleDeg= range(0,360,30);  #select angles that are multiples of 30.
$angleRad= $angleDeg*(Pi/180); #convert the angle to radians
$version= rint(3);  #select a version number from {0,1,2} each version will be a different trig function
$trigFunction=switch($version, "sin" , "cos" , "tan");  #if version#=0 -> sin , if version#=1 -> cos, etc.
$answer=switch($version, sin($angleRad), cos($angleRad) , tan($angleRad));
$answerRound=decimal(2, $answer);

USING CONDTIONS

Occasionally certain values will cause problems when designing algorithms. In these instances we can use conditions to re-run the algorithm to produce different numbers.

For example:

$number1= range(1,10,1);
$number2= range(1,10,1);
condition: ne($number1,$number2);  #this will not allow number1 = number 2

IMPORTANT: Conditions are often very problematic for the system to handle, hence you should prioritize good programming practice ahead of implementing conditions.
The below code is occasionally very useful for generating linear factors while avoiding linear factors that have a common integer factor (ie. avoiding $2x + 6$, $6x - 9$, etc.).

```maple
$numA = \text{range}(-9,9,1);
$numB = \text{range}(-9,9,1);
\text{condition: eq( } \gcd(\numA,\numB) , 1) ; \# forces \numA and \numB to be relatively prime
```
THE MAPLE FUNCTION

The maple function calls out to a remote server hosting a session of the Maple algebra-system. For those that are not familiar with the Maple algebra-system it can be thought of as the world’s most powerful calculator.

To demonstrate the power of the algebra-system image you require the equation

\[ \frac{1}{R_t} = \frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3} + \frac{1}{R_4}, \]

of which you are provided with the data, \( R_t = 2, \ R_1 = 1.4, \ R_2 = 2 \cdot R_1, \ R_3 = R_1 + R_2, \) and are required to solve for variable \( R_4 \)? Manipulating this equation will take considerable time, however using mapleTA this can be done in a few seconds.

```maple
$Rt= 2;
$R1= 1.4;
$R2= 2*$R1;
$R3= $R1+$R2;
$answer= maple("solve((1/$Rt)=(1/$R1)+ (1/$R2)+ (1/$R3)+ (1/$R4),R4);");
```

NOTICE: As you can see the algebra-system has the power to save you a great deal of time as authoring questions no longer requires instructors to solve the questions.

USING A SINGLE CALL FOR MANY CALCULATIONS

IMPORTANT: Using several separate maple functions within one question is extremely taxing on system resources and student loading time. As a result it is recommended you generate a list of several calculations and call back what you need out of the list.

```maple
$distanceKMpHr=range(1 , 1000 , 1);
$callout=maple("milesPhr:=convert($distanceKMpHr, 'units', 'km'/'h' , 'miles'/'h');
nanoPmilsec:=convert($distanceKMpHr, 'units', 'km'/'h' , 'nm'/'ms');
furlongPmin:=convert($distanceKMpHr, 'units', 'km'/'h' , 'furlong'/'min');
[milesPhr,nanoPmilsec,furlongPmin];");
$milesPhr= switch(0 , $callout);
$nanoPmilli= switch(1 , $callout);
$furlongPmin= switch(2 , $callout);
```
ADDING PHASORS/VECTORS (INCLUDING CONVERTING DEGREES TO/FROM RADIANS)

$mag1=\text{range}(100, 1000, 100);$
$\text{angle1deg}=\text{range}(-360, 360, 30);$
$mag2=\text{range}(100, 1000, 100);$
$\text{angle2deg}=\text{range}(-360, 360, 30);$
$mag3=\text{range}(100, 1000, 100);$
$\text{angle3deg}=\text{range}(-360, 360, 30);$
$\text{callout}=\text{maple}(
\begin{array}{l}
\text{angle1:=convert(}$\text{angle1deg}^\text{degrees}$,\text{radians}): \\
\text{angle2:=convert(}$\text{angle2deg}^\text{degrees}$,\text{radians}): \\
\text{angle3:=convert(}$\text{angle3deg}^\text{degrees}$,\text{radians}): \\
\text{vector1:=polar(}$\text{mag1},\text{angle1})$: \\
\text{vector2:=polar(}$\text{mag2},\text{angle2})$: \\
\text{vector3:=polar(}$\text{mag3},\text{angle3})$: \\
\text{result:=evalc(vector1+vector2+vector3)}; \#\text{notice this line adds the phasors} \\
\text{resultMagnitude:=evalf(abs(result))}; \#\text{this line determines the magnitude} \\
\text{resultAngleRad:=evalf(argument(result))}; \#\text{this line determines the angle, phase, in radians} \\
\text{resultAngleDeg:=convert(resultAngleRad, 'units', 'radians', 'degrees')}:\\\n\{\text{resultMagnitude, resultAngleDeg}\}; \\
\end{array}
$);

\text{NOTICE: Notice adding phasors, using the algebra-system, without using x-y components or other methods is tremendously time-saving.}
The below algorithm generates the plot:

$v1 = \text{range}(50, 200, 10)$;  
$\text{angle1deg} = \text{range}(1, 359, 2)$;  # odd numbers to avoid angles on the axis  
$v2 = \text{range}(50, 200, 10)$;  
$\text{angle2deg} = \text{range}(1, 359, 2)$;  # odd numbers to avoid angles on the axis  
$\text{callout} = \text{maple}($
#vector 1 define
angle1rad := \text{convert($\text{angle1deg}$*degrees, radians)};  
\text{vector1} := \text{polar($v1$, angle1rad)};
#vector 2 define
angle2rad := \text{convert($\text{angle2deg}$*degrees, radians)};  
\text{vector2} := \text{polar($v2$, angle2rad)};

$\text{vector1} = \text{switch}(0, \text{callout})$;  
$\text{vector2} = \text{switch}(1, \text{callout})$;

#this will pull the items from the list in the $\text{callout}$
$\text{vector1}$;  
$\text{vector2}$;

#this will plot the vectors and generate the proper-scaled axis
$\text{plot} = \text{plotmaple}($
#find max magnitude to scale axis appropriately
mag := \text{max(abs($v1$), abs($v2$))};  
#convert the vectors to rectangular form to graph
vector1rect := \text{evalc($v1$)};  
V1 := \langle \text{Re(vector1rect)}, \text{Im(vector1rect)} \rangle;  
vector2rect := \text{evalc($v2$)};  
V2 := \langle \text{Re(vector2rect)}, \text{Im(vector2rect)} \rangle;

#generate the components of the graph
\text{zero} := \text{VectorCalculus:-PlotVector}(<\text{mag}, 0>, \text{color}=\text{black}, \text{tickmarks}=[0,0], \text{width}=0.02, \text{size}=[400,400]);
\text{ninetys} := \text{VectorCalculus:-PlotVector}(<0, \text{mag}>, \text{color}=\text{black}, \text{tickmarks}=[0,0], \text{width}=0.02, \text{size}=[400,400]);
\text{oneEightys} := \text{VectorCalculus:-PlotVector}(<-\text{mag}, 0>, \text{color}=\text{black}, \text{tickmarks}=[0,0], \text{width}=0.02, \text{size}=[400,400]);
\text{twoSeventys} := \text{VectorCalculus:-PlotVector}(<0, -\text{mag}>, \text{color}=\text{black}, \text{tickmarks}=[0,0], \text{width}=0.02, \text{size}=[400,400]);
\text{vector1} := \text{VectorCalculus:-PlotVector}($V1$, \text{legend}=\text{voltageA(}$\text{$v1$ Volts, } \theta =$\text{angle1deg}$\degree$));
\text{vector2} := \text{VectorCalculus:-PlotVector}($V2$, \text{legend}=\text{voltageB(}$\text{$v2$ Volts, } \theta =$\text{angle2deg}$\degree$));

#display all components on the same graph
\text{plots}[:\text{display}](\text{zero}, \text{ninetys}, \text{oneEightys}, \text{twoSeventys}, \text{vector1}, \text{vector2});$}
The below algorithm generates the graph:

```maple
$voltage$ = range(120, 240, 120);
$constant$ = 60;
$callout$ = plotmaple(
  a := plot($voltage$ * sin(t), t = 0..10, color = red, legend = voltage);
  b := plot($constant$, t = 0..10, color = red, legend = constant);
  plots[display](a, b, thickness = 2, legendstyle = [location = left], gridlines,
                 labels = [time, voltage])
);
```

The below algorithm generates the graph:

```maple
$point1$ = [-6, -8];
$point2$ = [-2, -6];
$point3$ = [1, -3];
$plot$ = plotmaple(
  a := plots[pointplot]($point1$, symbol = solidcircle, symbolsize = 15, color = red);
  b := plots[pointplot]($point2$, symbol = solidcircle, symbolsize = 15, color = red);
  c := plots[pointplot]($point3$, symbol = circle, symbolsize = 15, color = red);
  d := plot(0.5 * x - 5, x = -6..-2);
  e := plot((x+1)^2 - 7, x = -2..1);
  f := plot(-3, x = 1..10);
  plots[display]([a, b, c, d, e, f], view = [-10..10, -10..10], axis = [gridlines = [20, color = grey]])
);
```

The below algorithm generates the visualization (mean=0, sigma=1)

```maple
$plot$ = plotmaple("with(stats):
  plot(statevalf[pdf, normald[10, 1]], 5..15);"");
```
FORCING THE DISPLAY OF NATURAL MATHEMATICS

\$a=\text{range}(2,9,1); \\
\$b=\text{range}(2,9,1); \\
\$c=\text{range}(2,9,1); \\
\$d=\text{range}(2,9,1); \\
\$e=\text{range}(2,9,1); \\
\$callout=\text{maple}(
\text{answerA:=}\text{expand}(\$a*\$g*(\$b*\$g-\$c)*(\$d*\$g+\$e)); \\
[\text{latex(answerA,output=string)},\text{latex(factor(answerA),output=string)}]; \\
")

\$textA=\text{switch}(0,\$callout); \\
\$textB=\text{switch}(1,\$callout);

NOTICE: Encasing the brackets "\(\)" or "\[\]" around the output from the above code will display natural mathematical font. For example: \(\text{A}\)

WORKING WITH MATHEMATICAL FUNCTIONS

\$coeff=\text{range}(2,9,1); \\
\$callout=\text{maple}(
\text{f:=x->(\$coeff*x^3); #define f(x)=\$coeff*x^3} \\
\text{display:=latex(f(x),output=string);} \\
\text{answerA:=f(1): #calculate the value of f(1)} \\
\text{answerB:=f(2): #calculate the value of f(1)} \\
\text{answerC:=diff(f(x),x): #differentiate f(x) w.r.t x} \\
\text{answerD:=eval(answerC,x=2): #evaluate the derivative at x=2} \\
[\text{display,answerA,answerB,answerC,answerD}]; \\
")

\$display=\text{switch}(0,\$callout); \\
\$fAtOne=\text{switch}(1,\$callout); \\
\$fAtTwo=\text{switch}(2,\$callout); \\
\$fPrime=\text{switch}(3,\$callout); \\
\$fPrimeAtTwo=\text{switch}(4,\$callout);
LOOKUP VALUES IN A TABLE

Question Context:

Use the following table to answer the question below,

<table>
<thead>
<tr>
<th>PRESS. (GAUGE)</th>
<th>TEMP.</th>
<th>SPECIFIC VOLUME</th>
<th>SPECIFIC ENTHALPY</th>
</tr>
</thead>
<tbody>
<tr>
<td>kPaG</td>
<td>°C</td>
<td>V_f</td>
<td>V_g</td>
</tr>
<tr>
<td>0</td>
<td>99.97</td>
<td>0.0010434</td>
<td>1.673</td>
</tr>
<tr>
<td>20</td>
<td>105.10</td>
<td>0.0010475</td>
<td>1.414</td>
</tr>
<tr>
<td>50</td>
<td>111.61</td>
<td>0.0010529</td>
<td>1.150</td>
</tr>
<tr>
<td>100</td>
<td>120.42</td>
<td>0.0010607</td>
<td>0.8803</td>
</tr>
</tbody>
</table>

What is the value in row=$row and column=$column ?

Answer=  $valuePulled

Algorithm:

#this part of the code defines the data to put into the table,
#******hint, use a .csv file from MSexcel to save you time*****
$numOfRows=3;  #this number specifies the number of rows of data you have
$numOfColumns=7;  #this field is not actually needed, unless values are being pulled randomly
$row1="0.099.97.0.0010434,1.673,419.0,2257,2676";
$row2="20,105.10,0.0010475,1.414,446.6,2243,2684";
$row3="50,111.61,0.0010529,1.150,468.2,2225,2694";
$row4="100,120.42,0.0010607,0.8803,505.6,2201,2707";

#this indicates the row# and column# to pull the data from
$row=range(1 , $numOfRows , 1);
$column=range(1 , $numOfColumns , 1);

#this part of the code automatically generates the table from the above information and pulls the required table entry
$callout=maple("table:=Matrix([[\$row1],[\$row2],[\$row3],[\$row4]]); value:=table($row,$column); display:=(MathML[ExportPresentation](table)); [display,convert(value,string)]; ");
$table=switch(0 , $callout);
$valuePulled=switch(2 , $callout);